A MAC AND ONE-WAY FUNCTION BASED FILTERING SCHEME

ZHAO Jing-Guo[1], LUO Qing-Yun[1] and Liu Zhi-Xiong[2]*
[1]Department of Computer and Information Science, Hunan Institute of Technology, Hengyang 421002, Hunan, China
[2]Department of Computer Science and Technology, Changsha University, Changsha 410022, Hunan, China
Emails: jg_zhao@foxmail.com, hnpulqy@foxmail.com, 121471998@qq.com

Submitted: Sep. 4, 2015 *Accepted: Nov. 3, 2015* *Published: Dec. 1, 2015*

Abstract- In optical sensor networks, the adversary can inject false reports and old packets, both of which can cause false alarms and drain out the limited energy of the network. Existing security mechanisms can detect and filter out only false reports but not the old packets during forwarding. Furthermore, they cannot resist cooperative attacks. A MAC and one-way function based Filtering Scheme (MOFS) was presented. Each node distributes its key and initial hash value to some other nodes after deployment. A data report must carry the MACs and fresh hash values from t detecting nodes. The forwarding nodes check the logicality of the relative position of the detecting nodes carried in the report, the correctness of the MACs and hash values, and the freshness of these hash values. Analysis and simulation results show that MOFS can not only filter false reports and old packets simultaneously, but also performs well on compromise toleration.

Index terms*:* Optical sensor networks; false reports; out-dated packets; MAC; one-way function.

I. INTRODUCTION

Optical sensor networks (OSNs) are widely used in many important fields including military surveillance, habitat monitoring and health care [1]. The sensor nodes with limited resources are usually deployed in hostile environments. In such circumstances, the security of sensors is of essential importance. Once a node is compromised by the adversary, all the secret information would be disclosed and thus can be abused to launch false report injection attacks [2], or replayed data injection attacks [3], i.e., to inject outdated reports into sensor networks. Defending the aforementioned attacks in WSNs is very important, because these illegitimate data not only cause false alarms but also may drain out the constrained resources of the sensors.

In recent years, some researches focused on preventing such false data and replayed data injections [4-15]. The feature of them is to attach t MACs (Message Authentication Codes) or time stamp to each report, and rely on intermediate nodes to verify the correctness. These schemes work well when only little compromised nodes existed in the network, while if more than t nodes are compromised, the attacker can then exploiting the fetched secrets to fake out un-realizable false data reports. Moreover, the attacker can also inject outdated data into the network without being detected en-route.

As illustrated in Fig. 1, here we assume that the adversary has compromised five nodes A_1, … , A_5. When the security threshold is set to five and the attacker fake a false report R:(e, M_1, M_2, M_3, M_4, M_5), and then send it to the neighbor nodes, then R would not be filtered out. This is because all MACs included in the report are correct and thus cannot be detected. Moreover, if the attacker exploits the compromised node A_5 to inject the outdated report R_0 into the network, then the forwarding nodes are also not able to detect, which cause the waste of energy.

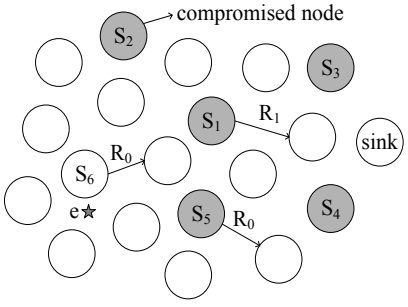

Fig. 1 False and outdated reports injections in WSN

In this paper, we presented a MAC and one-way function based filtering scheme named MOFS. The sensors use the function to generate a one-way hash chain, and then transmit the neighbor information and initial hash value to some intermediate nodes. Each report should carry the ID, hash value and MAC of *t* nodes which sensing the event simultaneously. The forwarding nodes then verify the logicality of the sensors' locations, the correctness of MACs, and both the correctness and freshness of the hash value, and thus to filter out false data and outdated data simultaneously.

II. RELATED WORK

Ye *et al.* firstly discussed about the problem of false data filtering in the Optical sensor networks, and presented a mechanism called SEF [3]. SEF divide the overall key pool into *n* partitions, and every partition includes *m* secret keys. Every node chooses a partition before deployment randomly, and then selects arbitrary *k* keys from the partition to store. After detecting the event, multiple nodes detecting the event simultaneously generate a report including *t* different MACs collaboratively. In the process of transmitting the data package, the intermediate node which owes the same key partition with the detection node can check up an MAC in the data package using a probability of k / m. All the false packages which sneak filtering would be filtered by sink finally. However, SEF has two severe problems. First, if a compromised node caches some legal data and injects a large number of copies into the network, then these copies would all be transmitted to sink, leading to a waste of energy. Second, if the attacker obtains *t* different key partitions arbitrarily, then he can fabricate false package which can't be distinguished.

Ma *et al.* put forward a sink verifying method, called REFE [4]. In the RSFS, all nodes and sink form the star topology, among which the cluster heads with stronger performance than the ordinary nodes gather data, and the gathering result should consist of all MACs produced by all detecting nodes. After accepting the gathering result, sink check the gathering result and the MACs, thereby filtering out the false data. This method isn't be restricted by safe threshold *t*, but false data can't be filtering out by transmitting nodes; therefore it isn't good for saving energy in the sensor network. In addition, some nodes dying or some new nodes joining the network would all lead to the change of the topology of the network, therefore the RSFS scheme, which only is suitable for the specific star topology, is not suitable for the actually deployed WSNs.

Yang *et al*. [5] proposed a statistical en-route filtering scheme based on multi-dimensional called MDSEF. Differ to the ordinary en-route filtering based on one dimension, the scheme divides an overall key pool into multiple sub-sets; each includes several key groups. After being deployed, each node can join a group in each set and randomly select some keys from these groups. This method can greatly improve the performance of covering probability and filtering effectiveness, without losing the resilience against nodes compromising. It then presented an axis-rotation based algorithm to associate multiple sets with the detecting terrain. Finally, a distributed stepwise refinement group joining algorithm is also applied for selecting groups. However, both the group joining algorithm and axis-rotation based algorithm lead to huge energy consumption for energy limited WSN.

Naresh *et al*. [6] pioneered an active filtering scheme for false data called AEFS to cope with false data injection attacks and DoS attacks simultaneously in WSN. Each node was firstly initialized with a hash chain. It then distributes its verifying key to some intermediate nodes. After sending a data report, the sensing nodes all immediately disclose their keys, enabling the forwarding nodes to check the corresponding reports. The scheme uses a so-called Hill Climbing algorithm to distributing keys with which the nodes lies closer to the source possess stronger filtering capacity than others. Furthermore, AEFS utilizes the actual property of broadcasting in wireless communications to deal with DoS attacks. The scheme is able to identify and filter out false data earlier with a low requirement in memory and computing. However, the Hill Climbing method also incurs a huge communication cost.

Bashir *et al*. [7] researched on the duplicate data elimination problem in hybrid RFID- sensor networks named RDFS. They thought that RFID data includes much duplication. The transmitting of RFID data through multiple hops toward sink will lead to extra energy consumption which is unnecessary and thus decreasing the network's lifecycle. The proposed scheme used a clustering mechanism and authorized cluster heads to eliminate duplicate data and forward filtered data towards the sink node. As a result, it can eliminate duplicated data effectively. Theoretical analysis and simulations illustrate that RDFS outperforms existing en-route filtering schemes on both filtering efficiency and energy consumption, but the adapted time synchronization technique isn't appropriate for wireless sensor network with limited energy.

Yang *et al*. [6] presented an en-route filtering scheme based on polynomials named PCREF. After deployed, all nodes in the scheme are grouped into some clusters. After that, according to a

pre-defined probability, each of them is assigned an authenticated polynomial and a checking polynomial. The authenticated polynomial is used to endorse the report and the checking polynomial is for verifying the accepted report, respectively. Different initiative polynomials will be adopted in different clusters. Compared with the aforementioned schemes which use MACs to endorse reports, polynomials based technique enables PCREF to achieve a better filtering capability.

Yang *et al*. [10] thought that the symmetric key technique is not safe enough for WSN and firstly tried to adopting asymmetric key technique on filtering false reports. The proposed CCEF scheme is based on commutative cipher, in which the source node establishes a secret association with sink, while all intermediate nodes using the witness key to verify the data reports without having to know the original session key. CCEF achieves a stronger protection against symmetric key based schemes. Ren *et al*. [12] proposed a LEDS scheme which exploiting the location-aware character of sensor networks to protect the security of data in WSN. The location-aware end-to-end security not only guarantees node-to-node verification through the package forwarding routes, but also can ensure an efficient en-route filtering capacity; however it is only suitable for some particular network routings.

To defending outdated data, Perrig *et al*. put forward the SNEP and μTESLA algorithms to check up outdated data in the SPINS protocol. μTESLA arithmetic assigns a hash function in advance for every node before deployment. Sink owes the whole information of all secrets. After detecting an event, the sensing nodes using the hash function to adores the sensing data, and send the result to the destination node. Both sides can update the counters and prepare transmission in the next time. Chen *et al*. proposed a scheme based on the time synchronization technique, TSPC [14]. Its basic ideology is that the side of sending inset time-poke information in every data package and sink distinguish outdated data through checking up the accuracy of time-poke.

Recent study indicates that the synchronization technique and public secret key technique have higher requirement on calculating and storage capability, and thus is difficult to apply to limited-performance wireless sensor network directly [2, 3]. However, the symmetrical secret key technique has many advantages, such as easy realization and low calculation complexity. It is possible the only one technique of data encryption which can be applied to sensor networks. Existing methods basing on technique of symmetrical secret key, such as SEF, RSF, MDSEF, RDFS, PCREF and so on, can't check up the freshness of the data, so they can't check outdated

package which is sent by compromise panel point. The deal such as SPINS, TSPC and so on can't check up and filter the outdated package in the transmission. This article mainly discusses how to filter out the false package and outdated package in the sensor network at the same time.

III. THE MOFS SCHEME

a. System model

Assume the sensors are distributed in a high density, such that each stimulus can be sensed by more than t sensor nodes simultaneously. They elect one of the nodes as the Center-of-Stimulus (CoS) [3] in a collaborative manner. We also assume that all detecting nodes belong to the neighbors of the CoS. The CoS then gathers t MACs and summarizes them to produce a data report. The data report is then forwarded toward the sink node through multiple hops.

The sink has strong self-protection, computation and storage capabilities, and possesses all secret information of the network, including the keys, hash values and relative positions of nodes. Sink can filter out all false and outdated reports that finally transmitted to.

We also assume that the network has a short time period of safe bootstrapping after deployed, during this period each node is safe to distribute its information without being compromised.

b. Deployment and bootstrapping

There is a global secret pool $G=\{K_i:0\leq i \leq W\text{-}1\}$, and every node A_i randomly selects a different key to store. In addition, we pre-assign each node a random data μ_i and a one-way function £ which has the features of one-way and irreversible, i.e., for a given input parameter a, it is easy to calculate $£(a)=b$, but it is impossible to introduce a from b [15].

Next, each node A_i can produce a one-way hash chain according to steps as follows. First, calculating the value of $£(x_i) = y_1$, $£^2(x_i) = £(y_1) = y_2,\ldots$, $£^u(x_i) = £(y_{u-1}) = y_u$ successively. Second, naming every value of above in reverse direction, i.e. commanding $h_i^u = y_1$, $h_i^{u-1} = y_2,\ldots$, $h_i^1 = y_u$, therefore we can obtain a one-way chain $H_i = h_i^1, h_i^2,\ldots, h_i^u$. In order to reduce expenses of storage, each node could only store a part of the hash values in the chain, such as no. k, no.$2k$…, and calculate the other values on the basis of equation 1 by these storing values.

$$h_i^j = £^{q_i - j}(h_i^{q_j}),\ \ j = 1, 2, \cdots, q_i - 1. \tag{1}$$

After deployment, every node A_i produces a data package $\{A_i, K_i, h_i^1\}$ including node ID, secret key and hash value which has the smallest index value, and broadcasts the package locally. After obtaining the broadcasting message, node A_i produces a note package: $\{A_i, A_{a1}, \ldots, A_{aj}, K_i, K_{a1}, \ldots, K_{aj}; h_i^1, h_{a1}^1, \ldots, h_{aj}^1\}$, where A_{aj} signifies the ID of A_i's neighbors, K_{aj} signifies the secret key stored in A_{aj}, and h_{aj}^1 signifies the hash value which has the smallest index value, respectively. Next, each node A_i builds a transmission path to sink, $Path(A_i) = \{A_i, A_1, \ldots, A_d, \text{sink}\}$, and sends the *note* message to sink. Finally, A_i deletes the hash value h_i^1 in the hash chain.

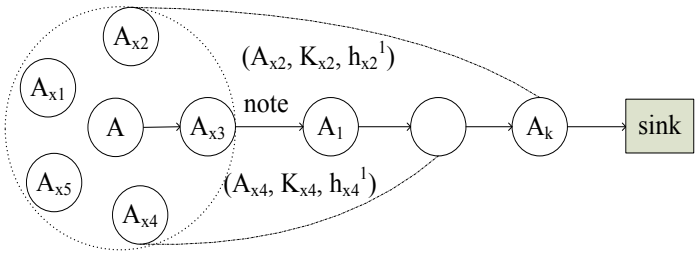

Fig. 2 Distribution of keys and hash values

After accepting the *note* package, node A_k ($1 \le k \le d$) selects out the neighbor information of A_i and stores it using a form as follows: $N(A_i)=\{A_{a1}, \ldots, A_{aj}\}$, and to be elected as a confirmation node of A_i using probability c_k / c_0, here c_k signifies the hops between A_k and sink, c_0 signifies the hops between A_0 and sink, respectively. If A_k is a confirmation node of A_i, it then selects a node A_{ax}($1 \le x \le j$) randomly from the *note* package and stores its information, then deletes the information from the *note* package. Otherwise, A_k forwards the *note* package directly. Fig. 2 provides the process of distributing secret keys and hash values.

c. Generation of reports

After an event occurred, all detecting nodes joining up to select a CoS. Cos will send its perception e to every detecting node. When accepting the data e, each detecting node will compare the perception value of itself to e. If the error is within some given threshold value, it then selects a secret key K_i to encrypt e and produces a message authentication code M_i:$K_i(e)$. Next, each detecting node sends its ID number, MAC, hash value which has the smallest index

number and index to Cos, and deletes the used hash value at the same time. Cos collects the information of another t-1 detecting nodes to produce a report. For example, assuming A_1, A_2, …, A_t are checking up an event e together, while A_1 is Cos, then the produced report is like this, R:{e; A_1, …, A_t; q_1, …, q_t; h_1^{q1}, …, h_t^{qt}; M_1, …, M_t}, here A_j signifies the node ID and q_j ($1 \le j \le t$) signifies the hash value index. Next, CoS will send the data report R to the node in the next forwarding hop.

d. Filtering on forwarding

After accepting the data package R, the intermediate node A_i first checks up whether the ID, hash value, number of MACs carried in the data package meet the requirements. It then judges the logic of all nodes' relative positions. At last, it checks up the correctness of MACs, the correctness and freshness of hash values, respectively. The detailed confirmation process is as follows:

Step 1: check up whether R including t node IDs, hash values, indexes and MACs.

Step 2: if not stored the neighbor information of A_1, abandoning R.

Step 3: check up whether every node A_2, … , A_t are neighbors of A_1.

Step 4: if stored the key K_v of certain node $A_v(1 \le v \le t)$, calculates M again through K_v and compares it to M_v. if M is equal to M_v ,abandoning R.

Step 5: if stored the hash value h_v^{qd} of certain node $A_v(1 \le v \le t)$, compares the index q_v to q_d . If q_v is less than q_d, signifies the hash value h_v^{qv} as un-fresh. Otherwise, judges whether h_v^{qd} is equal to $£^{qv\text{-}qd}(h_v^{qv})$, if h_v^{qd} is equal to $£^{qv\text{-}qd}(h_v^{qv})$, displaces the stored hash value h_v^{qd} of h_v^{qv}.

Step 6: if all of the above confirmations are successful, transmits R to the next hop.

Fig. 3 illustrates the pseudo-code of filtering on forwarding.

/* **Upon receiving a data report** *R* */

1. Check that t $\{A_v, M_v, h_v^{qv}\}$ tuples exist in R.
2. Check from the pre-stored neighbor information table, if it has not stored the neighbor information for A_1, drop R otherwise.
3. Check the t *-1* node IDs $\{A_v, 2 \le v \le t\}$ all belong to the neighbor set of A_1:N(A_1).
4. If it has one key $K \in \{K_v, 1 \le v \le t\}$,it computes $M=K(e)$ and see if the corresponding M_v is the same as M.
5. If it has stored one hash value h_v^{qd}, drop R if $q_v \le q_d$ is true. Otherwise it then checks if $h_v^{qd} = \pounds^{qv-qd}(h_v^{qv})$ is true, if true, then it updates the stored hash value to h_v^{qd}.
6. Forward R.

Fig.3 the pseudo-code of en-route filtering.

e. Example of en-route filtering

As illustrated in Fig.4, we assume that the transmitting path from a certain source A_1 to sink is denoted as: $Path(A_1)=\{A_1, A_6, A_7, A_8, \text{sink}\}$, where $A_2, \ldots, A_5$ are all neighbors of A_1. We further assume that A_6 owns the key K_1 and hash value h_1^1 for A_1, while node A_7 has the key K_2 and hash value h_2^1 for A_2. When t is equal to 5, we illustrate the procedure of verifying outdated and false reports.

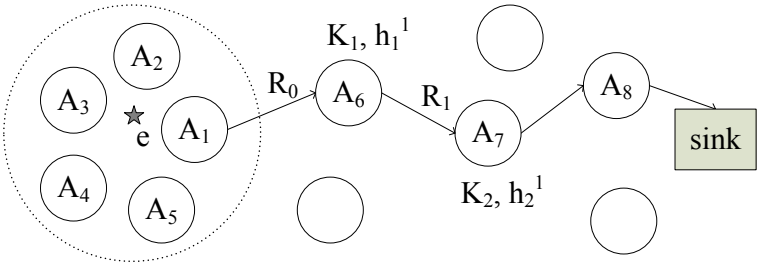

Fig. 4 an example of en-route filtering

We first discuss the verification of legitimate reports. Assuming a data report R_0 for an event e denoted by $\{e; A_1, \ldots, A_5; q_1, \ldots, q_5; h_1^2, \ldots, h_5^2; M_1, \ldots, M_5\}$, is generated by A_1 combining with other detecting nodes $A_2, \cdots, A_5$. R_0 is then sent to node A_6. After accepting R_0, A_6 checks whether

the numbers of node IDs, hash values, the corresponding indices, and 5 MACs are all 5 in the report. Actually, each intermediate node has to execute the above mentioned verification step and it is going to be omitted it in later text. It then examines if the index of the included hash value h_1^2 is larger than the pre-stored one h_1^1. It also checks whether the equation $h_1^1=F(h_1^2)$ is true. Furthermore, it re-produces a MAC using the stored K_1 and compares the result with the corresponding one in the report. If all the above verification processes are successful, R_0 is then sent to A_7. The hash value stored in node A_6 is updated to h_1^2accordingly. In the later phases, nodes S_7 and S_8 execute the verification procedure similarly, and update the stored hash value in node S_7 to h_2^2. Finally, R_0 is successfully passed to sink.

We then depict the authentication of outdated reports. We assume some compromised node A_6 is about to transmit a cached report R_0 to A_7. As A_7 has already updated its stored hash value to h_2^2 in the last verification phase, it is able to detect h_2^2 included in R_0 is un-fresh, and thus to drop R_0.

We finally talk about the verification of false data. We assume A_1、A_3、A_4、A_5 are compromised and are abused to fake a false report R_1which contains the faked MAC and hash value of A_2. Obviously, when A_7 receives the package, it can find that the included M_2 and h_2^2 are all faked because it had already pre-stored the key and hash value of A_2. Finally, R_1 is dropped by A_7.

VI. CONCLUSIONS PERFORMANCE ANALYSIS AND SIMULATIONS

a. The capability of anti-compromise

In the existing schemes such as SEF, the intermediate nodes only check up validity of the attached MACs in the data package, so after attacking arbitrarily t nodes (e.g., A_1，…，A_5 in Fig. 5), the adversary is able to fake unrecognized false package. While MOFS judges whether every node belonging to the neighbors of CoS to check up the logic of all detecting nodes' positions, thus is able to prevent nodes from different areas to fake package together in a collaborative manner. For example, assuming the attacker has compromised nodes from different areas A_1，…，A_5, and abused A_1 to fake a false package R:{E; A_1，A_2，…，A_5; M_1，M_2，…，M_5}, and further to send it into the network. After accepting R, if stored the neighbor information of A_1 , the forwarding node can judge that A_2 is not the neighbor of A_1 and

abandoning R. Otherwise, if had not stored the neighbor information of A_1, the forwarding node signifies that R isn't a package produced by the correct source node and also to abandon it accordingly.

Theorem 1: in the existing filtering schemes (e.g., SEF, REFS, MDSEF, RDFS, PCREF and so on), assuming the attacker has compromised N_c nodes randomly ($N_c \geq t$) in the network, then the probability for obtaining at least t different key partitions, can be denoted as,

$$P_S = \sum_{i=t}^{n} \{C(n,i) \times \sum_{j=0}^{i} [(-1)^j \times C_i^{i-j} \times (i-j)^{N_c}]\} / (n^{N_c}) \tag{2}$$

Proof: commanding N_c compromised nodes form a set Q_1, the number of methods of selecting t partitions from n ones is $C(n, t)$, and also these selected partitions form a set Q_2 . Next, the number of methods which set Q_2 form surjection to set Q_1 is,

$$\begin{aligned} \varphi &= C(t,t) \cdot t^{N_c} - C(t,t-1) \cdot (t-1)^{N_c} + C(t,t-2) \cdot (t-2)^{N_c} - \cdots \\ &\quad + (-1)^{t-2} \cdot C(t,2) \cdot 2^{N_c} + (-1)^{t-1} \cdot C(t,1) \cdot 1^{N_c} \\ &= \sum_{j=0}^{t-1} ((-1)^j \cdot C(t,t-j) \cdot (t-j)^{N_c}) \\ &= \sum_{j=0}^{t} ((-1)^j \cdot C(t,t-j) \cdot (t-j)^{N_c}) \end{aligned} \tag{3}$$

The number of methods that the attacker just obtains t key partitions after randomly compromising N_c nodes in the network is $C(n, t) \times \varphi$, similarly, we can calculate the number of methods that the attacker just obtains t +1, t +2,…, n key partitions, so the number of methods that the attacker obtaining more than t partitions is,

$$\sum_{i=t}^{n} \left[C(n,i) \cdot \sum_{j=0}^{i} ((-1)^j \cdot C(i,i-j) \cdot (i-j)^{N_c}) \right] \tag{4}$$

Obviously, the number of methods that each element in set Q_1 has an image in set Q_2 is n^{Nc}. Proved.

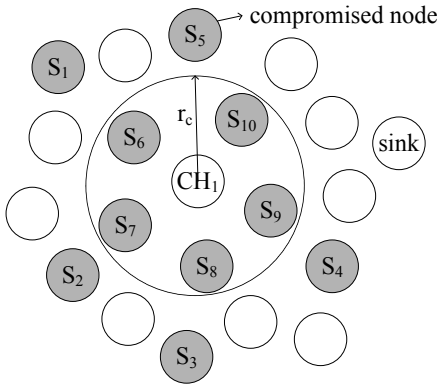

Fig. 5 Comparison of SEF and MOFS's compromise tolerance

In MOFS, in order to fake unrecognized false package, the attacker has to obtain more than t nodes, where some node A in it, making the other compromised nodes to be neighbors of A. For example, when t is equal to 5, after compromising A_6 and its neighbors A_7, ⋯, A_{10} in Fig. 5, the adversary is able to fake a false package with the Cos A_6 and additional 5 correct MACs through A_6, ⋯， A_{10}, finally, all forwarding nodes and sink can't filter out the false package.

Therom2: assuming the attacker obtains N_c ($N_c \geq$ t) nodes in the network randomly, the probability that existing more than t such nodes (there is some node A, making the other t-1 nodes are all neighbors of A) is,

$$P(\frac{r_c}{2}) < P_{MOFS} < P(r_c) \tag{5}$$

Here $p(\mu)$ is,

$$\sum_{i=t}^{N_c} \{C(N_c, i) \times (\pi\mu^2 / Z)^i \times [1-(\pi\mu^2 / Z)^i]^{N_c}\} \tag{6}$$

Proof: assuming that the area of the network is Z, and then every node has a probability $\pi\mu^2 / Z$ to distribute in the spherical zone M with a radius μ. Therefore, the probability of just existing t compromised nodes in the zone M is $p_b = C(N_c, t) \times (\pi\mu^2 / Z)^t \times (1- \pi\mu^2 / Z)^{Nc}$. Therefore, the probability of obtaining more than t nodes in the certain zone with a radius of μ is $p(\mu)$.

Making event g signifies this: among y ($t \leq y \leq N_c$) nodes, there is some node A, whose distances between the other y -1 nodes are all less than the communication radius r_c. Making event g_0 signifies this: y nodes belong to a same zone with a radius of $r_c/2$. Making event g_1 signifies this: y nodes belong to a same zone with a radius of r_c. Obviously, we can get g from g_0, and also can get g_0 from g_1. Proved.

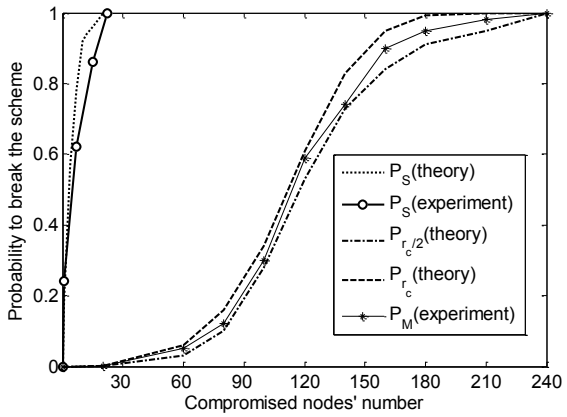

Fig. 6 Theoretic and experimental results of p_{SEF}、p_{MOFS}

Fig. 6 draws the theoretical analysis curve and simulation result of p_{SEF} and p_{MOFS}, here t is set to 5, r_c is 2.5 meters, n is equal to 15 and the network radius is set to 25 meters (simulation result is the average value from 2000 randomly tests under the same parameter settings). As is shown in Fig. 6, after obtaining little nodes, the attacker is able to break down SEF in a higher probability. However, in order to break down MOFS, it needs more compromised nodes. For example, when N_c is equal to 20, the probability that the attacker breaking down SEF is 98.9%, for MOFS, the probability is 0.02%. Therefore, from theoretical analysis and experiment result analysis, we can know that the compromise tolerate ability of MOFS is far better than SEF.

b. Filtering probability

Firstly, we analyze the ability of filtering false data in MOFS. Assuming the attacker obtains certain node A and A's N_d -1 neighbors. If N_d is more than t, the attacker can randomly forge unrecognized false packages. If N_d is less than t, the attacker must fake MACs and hash values of t - N_d neighbors of A. Because node A_i stores key and hash value of a neighbor node of the source node A with probability c_i / c_0, so the probability that the attacker can check up one of these t - N_d nodes is,

$$p_{b\_i} = \frac{t - N_d}{num(S) + 1} \cdot \frac{c_0 - i}{c_0} \tag{7}$$

The probability that the false package is filtered within H hops is,

$$p_b(H) = 1 - \prod_{j=1}^{H} (1 - \frac{t - N_d}{num(S) + 1} \cdot \frac{c_0 - j}{c_0}) \quad (8)$$

We then analyze the ability of filtering outdated data in MOFS. Assuming the compromised node A injects an outdated data R_a into the network. Making the number of neighbor nodes of A is $num(A)$, the number of hops from A to sink is c_0, the number of hops from A_i to sink is $c_i (1 \leq i \leq d)$. Obviously, c_i is equal to c_0 minus i .Therefore, the probability that the intermediate node A_i can check up the hash value of one out of these t nodes in the outdated package is,

$$p_{a\_i} = \frac{t}{num(S) + 1} \cdot \frac{c_0 - i}{c_0} \quad (9)$$

Since every forwarding node can filter R_a with a probability p_{a_i} , the probability that the outdated package being filtered within H hops is,

$$p_a(H) = 1 - \prod_{j=1}^{H} (1 - p_{a\_j}) = 1 - \prod_{j=1}^{H} (1 - \frac{t}{num(S) + 1} \cdot \frac{c_0 - j}{c_0}) \quad (10)$$

As is shown in Fig. 7 which draws the curves that both of outmoded package and false package changing according to the transmission hops H, (N_d=3, c_0= *20*, $num(S)$ =8， t =5). We can know from Fig. 7, MOFS can filter outdated package and false package with a higher probability simultaneously. For example, the proportions those within the first 5 hops to filter outdated package and fake package are 96.3% and 64%, respectively. Along with the increasing of the transmission hops, the filtering proportion is getting bigger and bigger, (all outdated package is filtered in the first 8 hops, and about 95% of the false package is filtered in the first 20 hops).

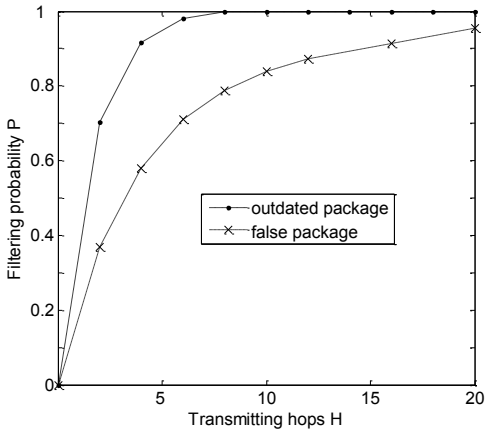

Fig. 7 The probability of MOFS to filter false and outdated package

c. Energy consumption

The energy which is used up by MOFS chiefly comes from: (1) in the initial stage of deployment, the expense that each node selects neighbor information, builds the transmission path to sink and distributes neighbor information to intermediate nodes; (2) the expense that transmission nodes check up MACs and hash values; (3) the expense that intermediate nodes transmit data package. In stage 1, interactive data package among nodes is short and lasts only little time. While in stage 2, the expense of MAC and hash value calculating is far less than data transmission. As a result, the expense of both stage 1 and stage 2 could be ignored.

Making L_r signifies the length of data package without adopting any safe mechanism, L_n signifies the length of node ID, L_i signifies the length of the key's index, L_k signifies the length of hash value's index and L_s signifies the length of the Bloom Filter. Then the length of data package in the MOFS and SEF respectively could be signified by $L_{r0} = L_r + 2L_s + (L_n + L_k)t$, $L_{rl} = L_r + L_s + tL_i$. For example, when L_r is 24bytes, L_n is 10bits, L_i is 10bits, L_k is 10bits and L_s is 64bits, L_{r0} and L_r is 52.45bytes and 38.27bytes, respectively. Obviously, comparing to SEF, the extra load in data package of MOFS will lead to the increase of transmission energy, but given to the capability that MOFS is able to filter out false and outdated data, the extra expense above could be tolerated. In addition, if the attacker injects false and outdated data into the network, comparing to SEF, MOFS can save more energy through filtering them as soon as possible. And we will check up it in the partition of simulation experiment. The equation behind illustrates the expense of transmitting"1 false data + β outdated data" for H hops:

$$E_0 = L_{r_0} \cdot [(1 + \sum_{i=2}^{H} (i \cdot p_{b\_i} \cdot \prod_{j=1}^{i-1} (1 - p_{b\_j})) + \beta \cdot \sum_{i=2}^{H} (i \cdot p_{a\_i} \cdot \prod_{j=1}^{i-1} (1 - p_{a\_j}))] \quad (11)$$

d. Expense of storage

In MOFS, each node needs to store a pre-distributed key, a one-way hash chain with length u, all upstream nodes' neighbor information and keys and hash values for part of the upstream nodes. For example, assuming a network with an area of $60 \times 60 m^2$, where randomly deploying 500 nodes whose radius are all 2.3m. Then the average number of neighbor nodes of a node is 7 and the average number of paths is 38. When the length of a key is 64bits, length of node ID is 10

bits, length of hash value is 64bits and the corresponding index is 19 bits, it requires a storage need of 2.1KB. As the mainstream nodes (e.g., the MICA2 node developed by UCB) equips more than 3 KB SRAM and 128KB ROM, and thus can meet the requirements obviously.

e. Simulation experiment

We build the simulated platform through C language, and the results are averaged from 10 simulation experiments. The simulation environment is as follows: in a $50\times50m^2$ square network area, randomly distributes 500 sensors, a static source node and a sink node are located in the sides of the area, respectively; the expense of transmitting and accepting data package of SEF is 60mW and 12mW respectively, and the expense of transmitting and accepting a MOFS data package is 81mW, 16mW, respectively; the communication radius and perception radius are 2.5m and 5m respectively.

As is shown in Fig. 8, we can know the change of filtering probability along with the transmission hops *H* when the number of compromise nodes is 15. As is shown below:

(1)MOFS could filter false and outdated packages with higher probabilities, e.g., the filtering proportion in the first 5 hops of the formal reach 80.3% and 96.7% respectively; along with the increasing of transmission hops, the probability of filtering false and outdated packages also increase quickly, being able to filter all false package within 19 hops and filter all outdated packages within 6 hops.

(2) SEF can filter about 25% false packages in the first 5 hops , e.g., being able to filter about 70% false packages in the first 20 hops. Because data package in the MOFS is attached with hash values, and thus is able to filter the outdated packages through checking up the freshness of them. However, SEF doesn't attach any "time variable parameters" in the data package, so can not filter the outdated packages.

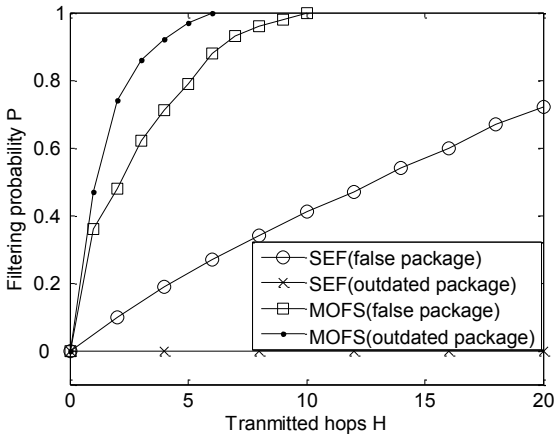

Fig. 8 Filtering probability P changes according to H

As is shown in Fig. 9, we can know the change of data transmission expense along with the transmitting hops H when the number of compromise nodes N_c is equal to 15. As is shown below:
(1) expense of SEF transmitting 100 false packages magnify with the increasing of transmitting hops, e.g., the expense of the first 7 transmitting hops is 3.34Joules and the expense of the first 20 transmitting hops is 9.48Joules; while at the same condition, the expense in the MOFS is much less, e.g., the expense of the first 6 transmitting hops is 1.49Joules, and after transmitting for 6 hops, the expense don't magnify any more because all false packages are filtered.
(2) The expense of SEF transmitting 100 outdated packages magnify with the increasing of transmitting hops. For example, when transmitting 20 hops, E is equal to 14.5Joules. However, at the same conditions, transmitting more than 6 hops, E is equal to 2.2Joules in the MOFS.

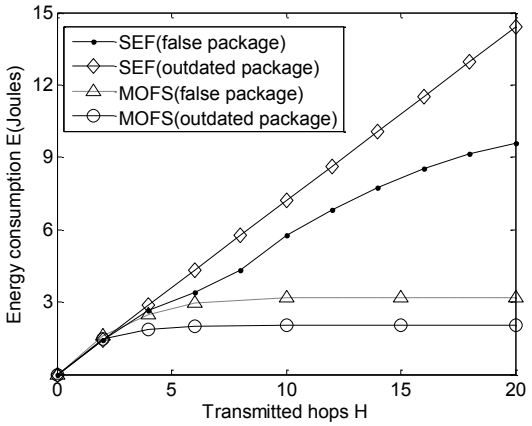

Fig. 9 E changes according to H

V. CONCLUSIONS

False and outdated data will incur false alarms and waste the limited energy in the network. This paper proposed a MAC and one-way function based scheme to defend them simultaneously. The novel solution is to bind the keys of nodes to their locations, and check the legitimacy of the locations of the endorsing nodes, and each node distributes its key and initial hash value to some other nodes after deployment. When a report is generated for an observed event, it must carry the MACs and fresh hash values from t detecting nodes. During forwarding, each node checks the legitimacy of the relative position of the detecting nodes carried in the report, the correctness of the MACs and hash values, and the freshness of these hash values, respectively. As for future work, we plan to extend our research results to mobile sink or multiple sinks situations.

AKNOWLEDGEMENT

This work was financially supported by Hunan province science and technology plan projects (2013FJ3095), The Project Development Plan of Science and Technology of Hengyang City (No. 2014KG63, 2014KG38), the Hunan Province College Research Project of the Teaching Reform (Xiang Jiao Tong [2015]291-562).

REFERENCES

[1] Talukdar S N, Apt J, Ilic M, et al., “Cascading Failures: Survival versus Prevention”, Electricity Journal, Vol.16, No.9, 2003, pp. 25-31.

[2] White D, Roschelle A, Peterson P, et al., “The 2003 Blackout: Solutions that Won't Cost a Fortune”, Electricity Journal, Vol.16, No.9, 2003, pp. 43-53.

[3] Leung E. Surge protection for power grids. IEEE Spectrum, 1997, 34-34.

[4] Ibrahim E S. “Electromagnetic fault current limiter”, Electric Power Systems Research, Vol.42, No.3, 1997, pp. 189–194.

[5] V.H, Porter J W., “Fault Current Limiters an Overview of EPRI Research”, IEEE Transactions on Power Apparatus & Systems, pas-99, No.5, 1980, pp.1964-1969.

[6] Onishi T, Aizawa N, Yamagata A, et al., "Stability of a shorted Nb3Sn coil cooled by a refrigerator for a magnetic shield type fault current limiter", IEEE Transactions on Applied Superconductivity, Vol.10, No.1, 2000, pp. 845-848.

[7] Keilin V, Kovalev I, Kruglov S, et al., "Model of HTS three-phase saturated core fault current limiter", IEEE Transactions on Applied Superconductivity, Vol.10, No.1, 2000, pp. 836-839

[8] Mukhopadhyay S C., "Synthesis and implementation of magnetic current limiter", Doctor of Engineering thesis, Faculty of Engineering, Kanazawa University, Japan, March Vol. 2000, 6, pp. 294-298.

[9] Dawson F P, Mukhopadhyay F P, Iwahara M, et al., "Analysis, design and experimental results for a passive current limiting device", Electric Power Applications, IEE Proceedings -, Vol.146, No.3, 1999,pp.309 - 316.

[10] Iwahara M, Mukhopadhyay S C, Fujiwara N, et al., "Development of passive fault current limiter in parallel biasing mode", Magnetics IEEE Transactions on, Vol. 35, No.5, 1999, pp. 3523-3525.

[11] Mukhopadhyay S C, Dawson F P, Iwahara M, et al., "A novel compact magnetic current limiter for three phase application", Magnetics IEEE Transactions on, Vol.36, No. 5, 2000, pp. 3568-3570.

[12] Malkin P, Klaus D., "Cap that current", Iee Review, Vol.47, No.2, 2001, pp. 41-45.

[13] Paul W, Lakner M, Rhyner J, et al., "Test of 1.2MVA High-T_c Superconducting Fault Current Limiter", Superconductor Science & Technology, 1997, Vol.10, pp. 914-918.

[14] Kim J T, Kim W S, Kim S H, et al., "Analysis of AC losses in HTS pancake windings for transformer according to the operating temperature", IEEE Transactions on Magnetics, Vol.41, No.5, 2005, pp.1888-1891.

[15] Steurer M, Brechna H, Frohlich K., "A nitrogen gas cooled, hybrid, high temperature superconducting fault current limiter", Applied Superconductivity IEEE Transactions on, Vol.10, No.1, 2000, pp. 840-844.

[16] Jiang Xu, Huanyan Qian, Wenhao Ying, et al., "A deployment algorithm for mobile wireless sensor networks based on the electrostatic field theory", The International Journal on Smart Sensing and Intelligent Systems, Vol.8, No.5, 2015, pp. 516-537.

[17] Qiuchan Bai, Chunxia Jin., "Image fusion and recognition based on compressed sensing theory", The International Journal on Smart Sensing and Intelligent Systems, Vol.8, No.5, 2015, pp. 159-180.

[18] Junfeng Qiao, Sanyang Liu, Xiaogang Qi and Gengzhong Zheng, "Transmission power control in wireless sensor networks under the minimum connected average node degree constraint", The International Journal on Smart Sensing and Intelligent Systems, Vol.8, No. 5, 2015, pp.801-821.